charles river analytics

Charles River Analytics Inc.
625 Mount Auburn St.
Cambridge, MA 02138

www.cra.com

p: 617.491.3474
f: 617.868.0780

15 JUN 2011

Reference: Government Contract No. N00014-09-C-0050, “Enhancing Simulation-based Training Adversary Tactics via Evolution (ESTATE)”
Charles River Analytics Contract No. C08098

Subject: Contractor’s Status Report: Quarterly Status Report #10
Reporting Dates: 3/15/2011 – 6/15/2011

Dear Dr. Hawkins,

The following is the Contractor’s Quarterly Status Report for the subject contract for the indicated period. During this reporting period we have concentrated on Task 3: Enhance Adaptation Techniques, Task 4: Develop Trainee Model Processing, Task 6: Simulation-based Training System Integration, and Task 7: Evaluation & Demonstration.

1. Summary of Progress

During the last reporting period, we had updated the ESTATE test environment (i.e. PROMPTER) to support third-party challenge generation and implement a testing framework to support simulated players, challenges, and performance. During this reporting period, we used this testing framework to evaluate different simulated player implementations and adaptive challenge generation components.

1.1 Player / Coevolution Testing Loop

In order to facilitate testing, we have implemented a modular system that can easily use multiple simulated player implementations and student-test coevolution components. For all testing and simulation discussed below, our methodology was as follows:

1. The simulated player is initialized with random values

2. The initial student and test populations within the coevolution algorithm are randomly generated

3. The coevolution algorithm is run for *N* generations

4. A challenge set consisting of 25 challenges is extracted from the coevolution algorithm (see section 1.2.3)

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **15 JUN 2011**	2. REPORT TYPE	3. DATES COVERED **00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE
Enhancing Simulation-Based Training Adversary Tactics via Evolution (ESTATE)

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Charles River Analytics Inc,625 Mount Auburn St,Cambridge,MA,02138

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**9**	

5. The player plays the challenge set, receiving feedback and learning as determined by its learning algorithm (see section 1.2.1). Logs are generated from the player's performance.

6. The student population within the coevolution algorithm is generated using a player model extracted from the player logs (see section 1.2.2). The test population is initialized using one of several methods (see section 1.2.3).

7. Steps 3 through 6 are repeated until the player has played 100 challenge sets.

The player's initial accuracy is measured, as well as its accuracy after every challenge set. Accuracy is measured without altering the player's associations and acts as an objective metric by which to evaluate. For any given combination of parameters, player implementation, and coevolution components, the above testing loop is run 20 times. For comparison, the same player implementation is tested 20 times against 100 randomly generated challenge sets (25 challenges per challenge set, and one definition per set.) For both coevolution and random, performance is averaged across all 20 test runs.

1.2 Player Simulation and Challenge Set Generation

1.2.1 Player Simulation

1.2.1.1 Initial 'Uniform Learner' Player Simulation

Last reporting period, we discussed how our simulated player uses an association matrix to model the associations between symbols and definitions, where a matrix value [*m*, *n*] represents the association that symbol *m* is the correct answer for definition *n*. Our initial simulated player's learning algorithm received feedback after every challenge. When the player learns from a challenge, the correct symbol's matrix value was incremented by a fixed value, while the three incorrect symbols had their values decremented by the same value.

The player was also given the ability to probabilistically fail to learn or to randomly "forget" a random entry within the matrix (i.e. by assigning a random value to that entry) after each challenge. Sensitivity analysis showed that while reducing the learning probability had a linear affect on the player's overall learning performance, the forgetting mechanism did not strongly affect performance until the forgetting probability became so high as to completely disrupt learning.

The player's time to answer is determined by the values within its association matrix. The challenge presents the player with a definition and four symbols that serve as potential answers. The player always chooses the symbol with the highest association for the definition, but the time to answer represents the player's confidence. The time to answer is determined by the following equation, where *minDistance* is the difference between the highest-ranked symbol for the definition and second-highest symbol.

$$time = \frac{1}{minDistance + 0.1}$$

As a result, the player’s answer time increases as the association values for the two highest symbols become closer, representing confusion of which is the correct answer.

While our initial player simulation modeled learning, it did not implement a zone of proximal development (ZPD). As a result, the player learned the same for all challenges regardless of difficulty. Testing showed that presenting the simulated player with challenges generated by student-test coevolution resulted in learning no better than if the player had been presented with randomly generated challenges (Figure 1). This illuminated the need to modify the simulated player’s learning model to more accurately represent this phenomenon that we would expect to see in a real human player. Only then would we have a better comparison for our adaptive challenge generation approach.

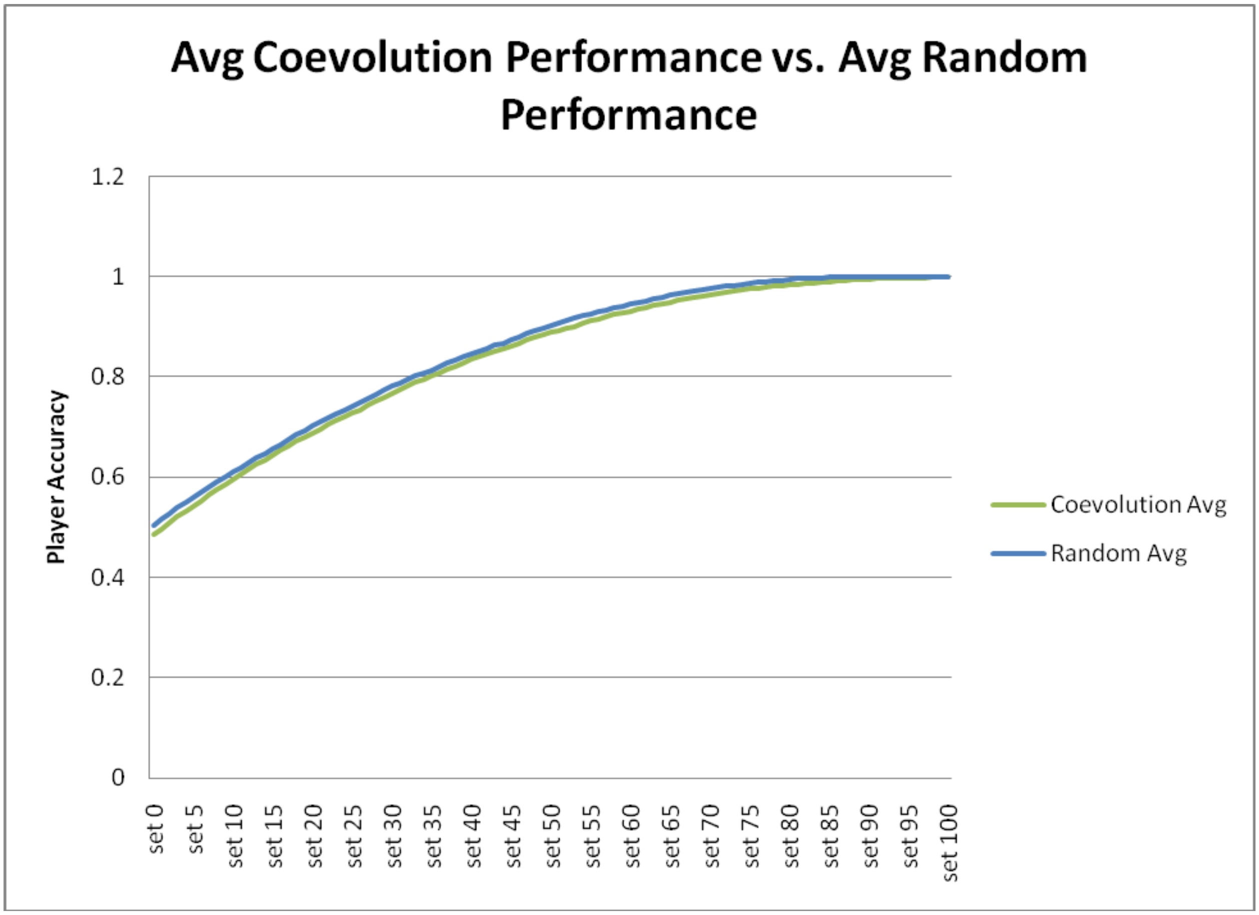

Figure 1 – Coevolution and Random Training Comparison for a Uniformly Learning Player

1.2.1.2 Implementing a Zone of Proximal Development

Our ZPD learner takes the same association matrix and learning algorithm as our initial uniform learner, with a few modifications. First, the ZPD learner player only learns when the player’s answer is incorrect, in order to prevent the player from learning from challenges that are too easy. Secondly, the ZPD learner only learns when it is able to correctly eliminate two of the symbols, narrowing the challenge down to a choice between the right answer and a single incorrect choice. This prevents the player from learning from challenges that are too difficult.

One final modification was made to the learning algorithm, as testing showed it was possible for a randomly initialized association matrix to suffer from deadlock, such that certain symbol/definition pairs could not be learned due to their values being too low. For instance, if the correct symbol for a given definition is randomly assigned an association value of 0, it is impossible for any incorrect answers to be eliminated, so no learning can occur. To address this issue, when a player receives feedback that symbol *m* is associated with definition *n*, all definitions other than *n* have their association value for *m* decremented.

As discussed in 1.2.3 below, our ZPD learning player exhibits superior learning when presented with student-test coevolution challenges than it does when presented with random challenges.

1.2.2 Player Model Extraction

Now that we have a model of a simulated player, our next focus was to be able to extract a model of this player based on player logs. This player model is then used later to seed our student population within the coevolution algorithm, representing the possible space of the player. The logs show a complete record of the player's past performance, including the challenge, the player's answer, the time it took the player to answer, and the date the challenge was performed.

To model the player via logs, we create directed graphs for each definition. Each node in the graph represents a symbol, and the directed links indicate player preference (i.e. the player chose symbol *a* over symbol *b* when they were both presented as choices in a challenge.) From these graphs, we can perform a topological sort to create complete orders for each definition. Using these orders, the student association matrices can be generated.

Our first step in player model extraction is to remove duplicate records. If the player has performed the exact same challenge more than once (the definition and all four symbols are identical), then only the newest record is considered.

Next, a directed graph is created for each definition. When links between symbols are created, the links store the date the challenge was played as well as the time it took the player to answer.

As the player learns, its preferences among symbols changes. Therefore, it is possible for cycles to be created within the graph. (For example, if a player initially chose symbol *a* over symbol *b*, but then, after learning, chose *b* over *a*, a cycle between the two nodes would be introduced.) Before performing the topological sort, all cycles are removed by removing the oldest links in the cycle, one at a time, until the cycle is broken.

With all cycles removed, it is possible to perform a topological sort on each graph. When a student is generated for the coevolution algorithm, the sort is used to create a complete order. The highest-ordered symbol is given a fixed value of 0.9 within the student's association matrix. All subsequent values are given random descending values, in order. The player's average time-to-answer is factored into the process, so that as the player answers more quickly, all symbols after the highest have their values forced down towards zero.

Since our simulated player also uses an association matrix, we can directly measure how well the player model extraction is working. To do so, we compare the simulated player's matrix to the student population's matrices. For each student, the absolute error is calculated on a per-definition basis. These comparisons are then averaged across all students, and then finally across all definitions.

Our initial player model extraction showed less than satisfying convergence, with the error increasing before decreasing, indicating that the student population was diverging, then converging. Figure 2 shows the average absolute error between students and the player after each challenge set is performed. As more challenges are played (resulting in more log data) the models converge on the player, but accuracy was not satisfactory.

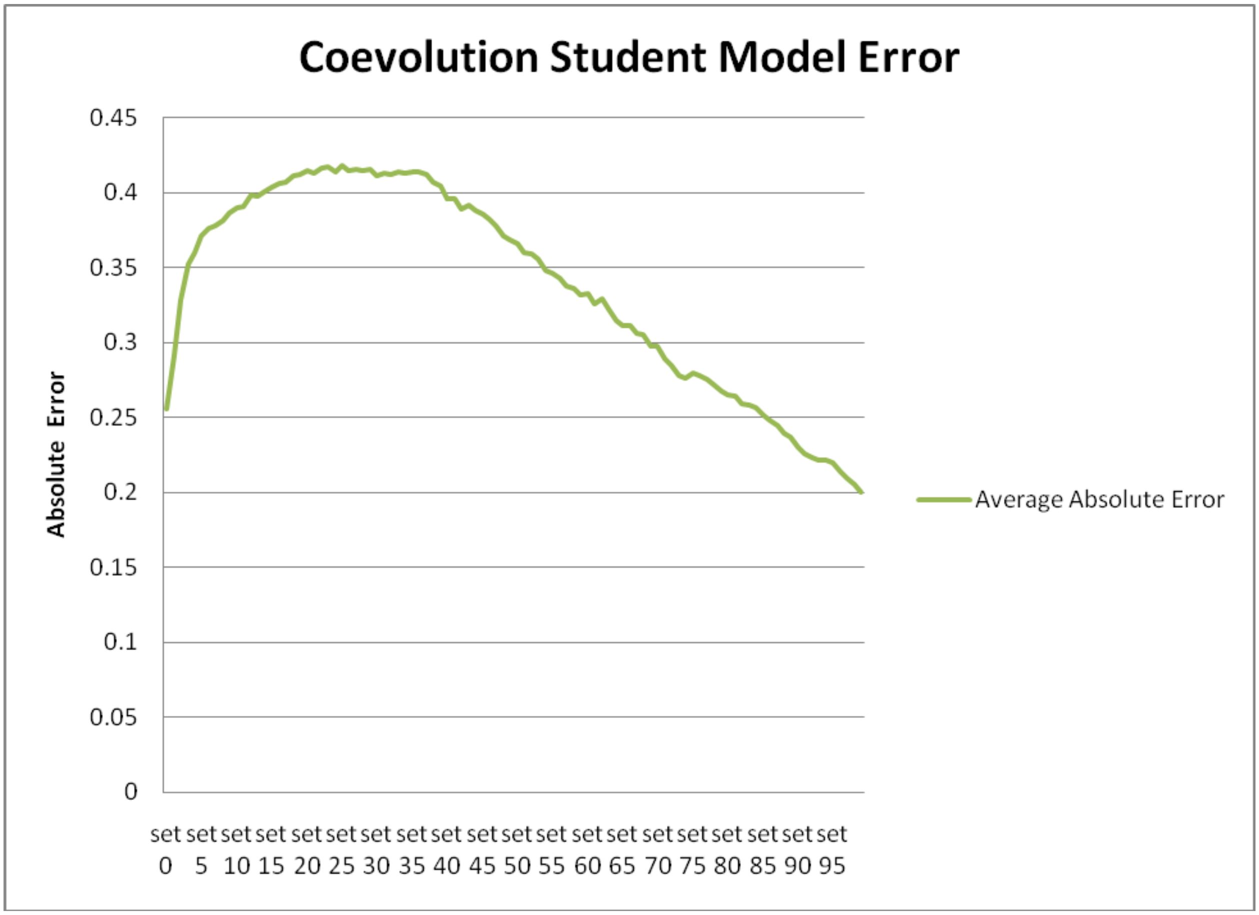

Figure 2 – Student Model Error for the Initial Player Model Extraction Algorithm

After reviewing the algorithm, we determined that our topological sort was insufficient. When performing a topological sort on a partial ordering graph, there are multiple complete orders that obey the constraints of the graph. However, our sort implementation was always returning the same complete ordering. So, while students had different values in their association matrices, the ordering of those values were identical.

The algorithm was modified to give each student a random complete order that obeyed the constraints of the graph. For, example, if a graph of symbols A,B,C,D,E,F determined that {A,B,C} < {D} < {E,F}, the initial implementation would always return the complete order A<B<C<D<E<F. There is no guarantee that this is the ordering within the player's matrix, of course. The modified algorithm can return multiple orders, such as B<C<A<D<F<E, C<A<B<D<E<F, etc. This had a drastic impact on overall accuracy, as seen in Figure 3, which shows a much greater reduction in overall error as logs are generated.

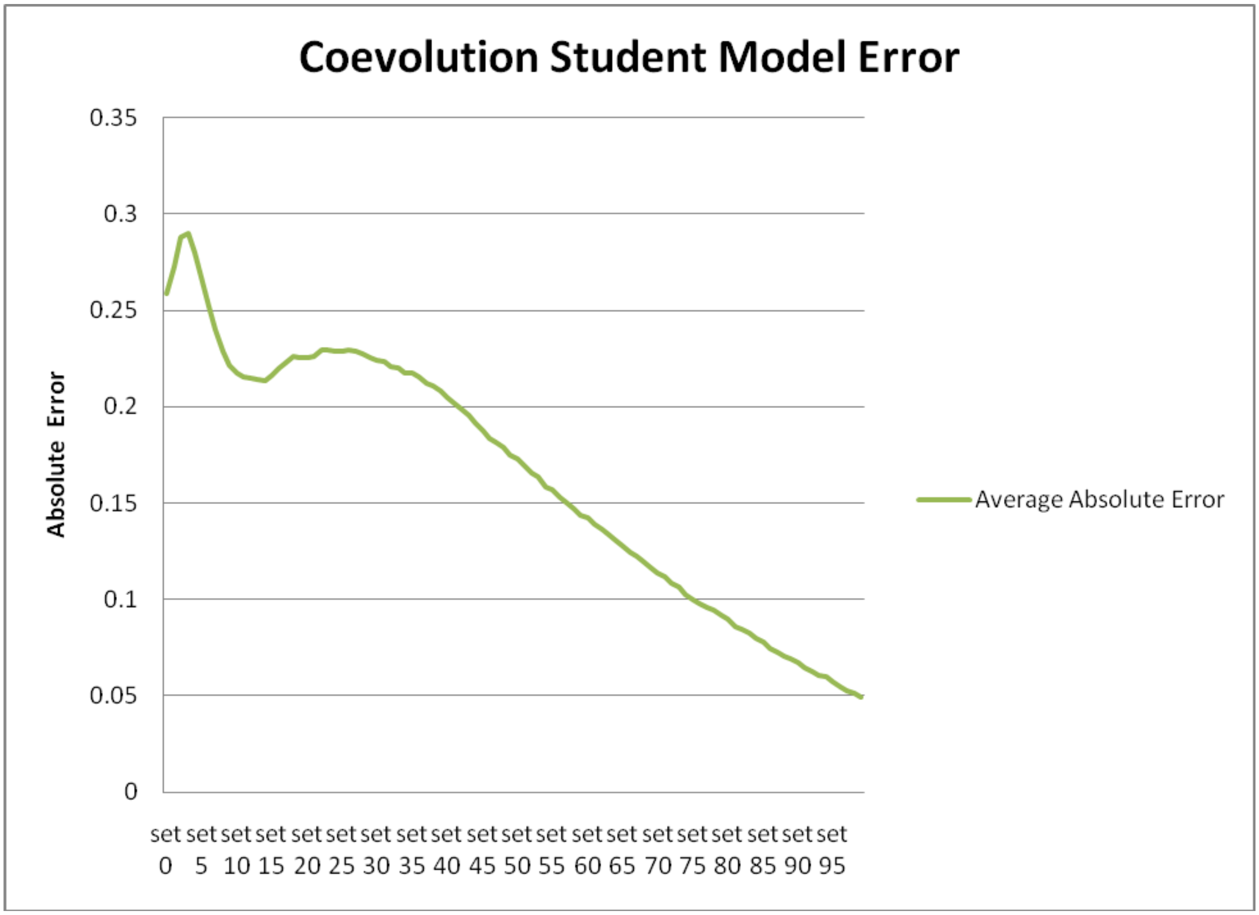

Figure 3 – Student Model Error for the Modified Player Model Extraction Algorithm

1.2.3 Challenge Set Generation

We have experimented with multiple methods of challenge set generation. All methods tested thus far create a pool of challenge candidates from tests that are dropped from the test population. Tests are dropped from the population when they are either defeated by all students, or when they are defeated by the same combination of students as another test.

Early experiments determined that tests should not be added to the candidate pool unless they have survived at least one generation without being defeated. Tests that are created and defeated during the same generation tend to be very uniform, addressing the same definition and symbols as their parent. (Since the mutation rate is kept low, some test offspring are identical to their parents.)

Our first method was to sample a range of dropped tests, sorted based upon when they were dropped from the test population. Our expectation was that tests were dropped primarily because they had been defeated by all students, so a chronological sample of defeated tests would provide a "ladder" of challenges that the simulated player could then follow to mirror the progress of the student population. However, analysis showed that almost no tests are dropped because they are defeated by all students. Instead, the vast majority of tests are dropped because they are defeated by an identical combination of students as another test. Experiments using this challenge set approach yielded results no better than random training.

Our next method was to sort defeated challenges by the number of students they defeated, and sample from a range of those scores. Using our ZPD learning player (see Section 1.2.1.2) we found that sampling tests that defeated 75-100 of the 100 students in the population yielded better results than random training (Figure 4). However, when using a uniform learner this increase in performance disappeared. We suspect that our sampling method is working primarily because tests that defeat 75-100 students tend to be hard for the player (ensuring it has a chance to learn) but not too hard (ensuring that it can eliminate two of the incorrect choices.) We feel that while this is a promising start, further work must be done to improve challenge set generation.

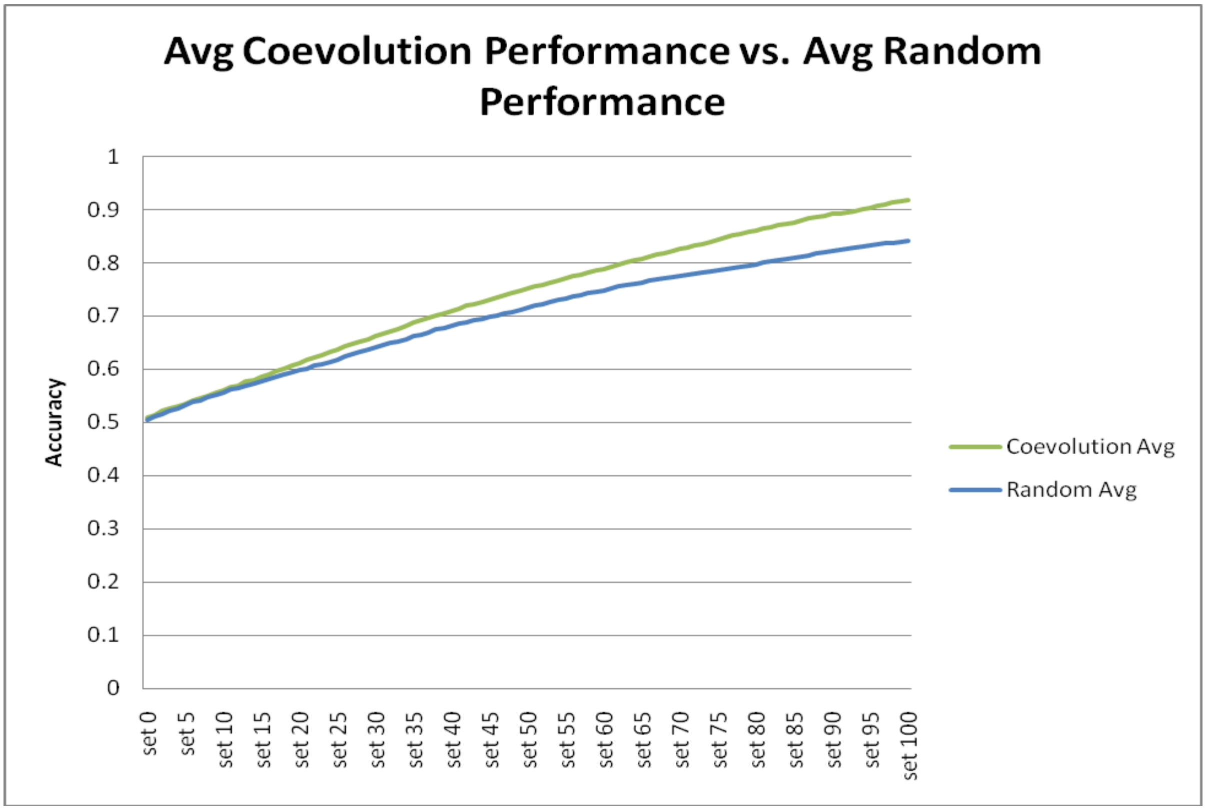

Figure 4 – Coevolution Performance When Sampling Tests Based on # of Students Defeated

DISTRUBITION A. Approved for public release; distribution is unlimited

1.3 2011 Annual Program Review

Finally, during the indicated period we presented our work over the past year at the Harold Hawkins ONR 341 Program Review FY11 in Arlington, VA on 3 JUN 2011. In attendance were Brad Rosenberg, Principal Investigator for the ESTATE effort, Dr. Hawkins, Program Manager for the effort, other representatives within the research and development community from academia, industry, and Government. Mr. Rosenberg provided an overview of the originating motivation of ESTATE, described its evolution to support adaptive challenge generation over adaptive adversaries, provided the overall approach, presented the path from single-dimensional skills models using item response theory through abstract games for more complex representations to applications to real-world training systems through the PROMPTER framework, discussed the most recent work in testing, and provided an overview of next steps. Key takeaways during discussion were: (1) what evidence exists that the ESTATE approach will scale to other domains and other types of training (e.g., procedural, decision-making), (2) comparison and issues that may develop with an ideal learner, and (3) abstracting the domain beyond training pictorial mnemonics.

2. Scheduled Items

In the next reporting period we plan to address the following items:

- Continue experimentation of the ESTATE approach within the PROMPTER framework with a focus on
  - Student population initialization strategies
  - Test population initialization strategies
  - Student-Test Coevolution algorithmic properties
  - Strategies for selecting tests to form challenge sets
- Draft a line of reasoning indicating why the ESTATE approach could scale to real-world training systems across domains
- Perform a comparative evaluation against other standard algorithms in addition to random, such as a standard curriculum and space repetition models
- Begin to apply the approach to other domains in training (1) declarative memory, (2) procedural memory, and (3) decision-making

Sincerely,

Brad Rosenberg
Principal Investigator